ROLL NUMBER : 2105890
NAME: HARSHIT SINGHANIA

1. WAP to find out the smallest and largest element stored in an array of n integers.

```c
#include <stdio.h>

int main()
{
    int a[50], i, n, large, small;
    printf("How many elements:");
    scanf("%d", &n);

    for (i = 0; i < n; i++)
    {
        printf("enter element %d:", i + 1);
        scanf("%d", &a[i]);
    }
    large = small = a[0];
    for (i = 1; i < n; ++i)
    {
        if (a[i] > large)
            large = a[i];
        if (a[i] < small)
            small = a[i];
    }
    printf("The largest element is %d", large);
    printf("\nThe smallest element is %d", small);

    return 0;
}
```

```
enter element 4:4
enter element 5:5
enter element 6:6
enter element 7:7
enter element 8:8
enter element 9:9
enter element 10:10
The largest element is 10
The smallest element is 1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

2. WAP to reverse the contents of an array of n elements.

```c
#include <stdio.h>

int main()
{
    int a[50], i, n;
    printf("How many elements:");
    scanf("%d", &n);

    for (i = 0; i < n; i++)
    {
        printf("enter element %d:", i + 1);
        scanf("%d", &a[i]);
    }
    printf("current array is \n" ) ;
    for (i = 0; i < n; i++)
    {
        printf("%d \n", a[i]);
    }
    printf("reversed array will be : \n");
    for (i = n - 1; i >= 0; i--)
    {
        printf("%d \n", a[i]);
    }
}
```

```
enter element 2:1
enter element 3:2
enter element 4:4
enter element 5:6
enter element 6:5
enter element 7:7
enter element 8:8
enter element 9:10
enter element 10:9
current array is
3
1
2
4
6
5
7
8
10
9
reversed array will be :
9
10
8
7
5
6
4
2
```

3. WAP to search an element in an array of n numbers.

```c
#include <stdio.h>

int search(int arr[], int n, int x)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == x)
            return i;
    return -1;
}

int main(void)
{
    int arr[] = {2, 3, 4, 10, 40};
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = search(arr, n, x);
    (result == -1)
        ? printf("Element is not present in array")
        : printf("Element is present at index %d", result);
```

```
    return 0;
}
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22> cd "c:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22\" ; if ($?) { gcc 3.c -o
3 } ; if ($?) { .\3 }
Element is present at index 3
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

4. WAP to sort an array of n numbers.

```c
#include <stdio.h>
void print(int a[], int n)
{
    int i;
    for (i = 0; i < n; i++)
    {
        printf("%d ", a[i]);
    }
}
void bubble(int a[], int n)
{
    int i, j, temp;
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (a[j] < a[i])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
}
void main()
{
    int i, j, temp;
```

```
    int a[5] = {10, 35, 32, 13, 26};
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting array elements are - \n");
    print(a, n);
    bubble(a, n);
    printf("\nAfter sorting array elements are - \n");
    print(a, n);
}
```

```
Before sorting array elements are -
10 35 32 13 26
After sorting array elements are -
10 13 26 32 35
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

5. Given an unsorted array of size n, WAP to find number of elements between two elements a and b (both inclusive).

   Input : arr = [1, 2, 2, 7, 5, 4], a=2 and b=5

   Output : 4

   (The numbers are: 2, 2, 5, 4)

   If a=6 b=15, then output will be 0

```
#include <stdio.h>

int main(void)
{
    int arr1[] = {1, 2, 2, 7, 5, 4};
    size_t n = sizeof(arr1) / sizeof(*arr1);

    int a = 2, b = 5;

    size_t count = 0;
    int lower_limit = 0, upper_limit = 0;
```

```
    for (size_t i = 0; i < n; i++)
    {
        if (a <= arr1[i] && arr1[i] <= b)
        {
            ++count;
            lower_limit |= arr1[i] == a;
            upper_limit |= arr1[i] == b;
        }
    }

    count += !lower_limit + !upper_limit;

    printf("Number of elements between %d and %d is %zu\nv", a, b, count);

    a = 6;
    b = 15;

    count = 0;
    lower_limit = 0;
    upper_limit = 0;

    for (size_t i = 0; i < n; i++)
    {
        if (a <= arr1[i] && arr1[i] <= b)
        {
            ++count;
            lower_limit |= arr1[i] == a;
            upper_limit |= arr1[i] == b;
        }
    }

    count += !lower_limit + !upper_limit;

    printf("Number of elements between %d and %d is %zu\n", a, b, count);

    return 0;
}
```

```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22> cd "c:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22\" ; if ($?) { gcc 5.c -o
5 } ; if ($?) { .\5 }
Number of elements between 2 and 5 is 4
vNumber of elements between 6 and 15 is 3
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

6. Given an array, WAP to print the next greater element (NGE) for every element. The next greater element for an element x is the first greater element on the right side of x in array. Elements for which no greater element exist, consider next greater element as -1.

Sample Input & Output

For the input array [2, 5, 3, 9, 7], the next greater elements for each element are as follows.

Element	NGE	Element	NGE
2	5	9	-1
5	9	7	-1
3	9		

```c
#include<stdio.h>

void printNGE(int arr[], int n)
{
    int next, i, j;
    for (i=0; i<n; i++)
    {
        next = -1;
        for (j = i+1; j<n; j++)
        {
            if (arr[i] < arr[j])
            {
                next = arr[j];
                break;
            }
        }
        printf("%d -- %d", arr[i], next);
        printf("\n");
    }
```

```
}

int main()
{
    int arr[]= {2,5,3,9,7};
    int n = sizeof(arr)/sizeof(arr[0]);
    printNGE(arr, n);
    return 0;
}
```

```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22> cd "c:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22\" ; if ($?) { gcc 6.c
6 } ; if ($?) { .\6 }
2 -- 5
5 -- 9
3 -- 9
9 -- -1
7 -- -1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

7. Let A be nXn square matrix array. WAP by using appropriate user defined functions for the following:

   a) Find the number of nonzero elements in A

   b) Find the sum of the elements above the leading diagonal.

   c) Display the elements below the minor diagonal.

   d) Find the product of the diagonal elements.

```
#include <stdio.h>

void count_zeros(int row, int col, int matrix[row][col]);
void sum_leading_zeros(int row, int col, int matrix[row][col]);
void product_diagonal_elements(int row, int col, int matrix[row][col]);
void display_elements_minor(int row, int col, int matrix[row][col]);

int main()
{
    int i, j;
    int rowm, colm;
    printf("Enter the number of rows: ");
    scanf("%d", &rowm);
```

```c
    printf("Enter the number of columns: ");
    scanf("%d", &colm);
    int matrixm[rowm][colm];
    for (i = 0; i < rowm; i++)
    {
        for (j = 0; j < colm; j++)
        {
            printf("Enter the element: ");
            scanf("%d", &matrixm[i][j]);
        }
    }
    count_zeros(rowm, colm, matrixm);
    sum_leading_zeros(rowm, colm, matrixm);
    product_diagonal_elements(rowm, colm, matrixm);
    display_elements_minor(rowm, colm, matrixm);
}

void count_zeros(int row, int col, int matrix[row][col])
{
    int i, j;
    int count = 0;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            if (matrix[i][j] != 0)
            {
                count++;
            }
        }
    }
    printf("%d\n", count);
}

void sum_leading_zeros(int row, int col, int matrix[row][col])
{
    int i, j;
    int sum = 0;
    for (i = 0; i < row; i++)
    {
```

```
        for (j = 0; j < col; j++)
        {
            if (j > 1)
            {
                sum += matrix[i][j];
            }
        }
    }
    printf("%d\n", sum);
}

void product_diagonal_elements(int row, int col, int matrix[row][col])
{
    int i, j;
    int product = 0;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            if (i == j)
            {
                product *= matrix[i][j];
            }
        }
    }
    printf("%d\n", product);
}

void display_elements_minor(int row, int col, int matrix[row][col])
{
    int i, j;
    for (i = 0; i < row; i++)
    {
        for (j = col - 1; j >= 0; j--)
        {
            if (i == j)
                printf("%d ", matrix[i][j]);
        }
    }
}
```

```
Enter the number of rows: 3
Enter the number of columns: 3
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 4
Enter the element: 5
Enter the element: 6
Enter the element: 7
Enter the element: 8
Enter the element: 9
9
18
0
1 5 9
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

8. Given an unsorted array arr[] and two numbers x and y, find the minimum distance between x and y in arr[]. The array might also contain duplicates. You may assume that both x and y are different and present in arr[].
   Input: arr[] = {3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3}, x = 3, y = 6
   Output: Minimum distance between 3 and 6 is 4.

```c
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

int minDist(int arr[], int n, int x, int y)
{
    int i, j;
    int min_dist = INT_MAX;
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if ((x == arr[i] && y == arr[j]
                || y == arr[i] && x == arr[j])
                && min_dist > abs(i - j)) {
                min_dist = abs(i - j);
```

```
            }
        }
    }
    if (min_dist > n) {
        return -1;
    }
    return min_dist;
}


int main()
{
    int arr[] = { 3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int x = 3;
    int y = 6;

    printf("Minimum distance between %d and %d is %d\n", x,
        y, minDist(arr, n, x, y));
    return 0;
}
```

```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22> cd "c:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22\" ; if ($?) { gcc te
eRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Minimum distance between 3 and 6 is 4
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

Home Assignments

1. WAP to find out the second smallest and second largest element stored in an array.

```
#include <stdio.h>

int main()
{
    int n;
```

```c
    printf("Enter the number of elements:");
    scanf("%d", &n);
    int a[n];
    for (int i = 0; i < n; i++)
    {
        printf("Enter the %d element: ", i+1);
        scanf("%d \n", &a[i]);
    }
    for (int i = 0; i < n; i++)
    {
        int temp;
        for (int j = i + 1; j < n; j++)
        {
            if (a[i] < a[j])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    printf("The second smallest element is %d", a[n - 2]);
    printf("\n");
    printf("The second largest element is %d", a[1]);
    return 0;
}
```

```
Enter the number of elements:3
Enter the 1 element: 1
2
Enter the 2 element: 2
Enter the 3 element: 3
The second smallest element is 2
The second largest element is 2
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

2. WAP to arrange the elements of an array such that all even numbers are followed by all odd numbers.

```c
#include <stdio.h>
#include <conio.h>

int main()
{
    int a[10000], b[10000], i, n, j, k, temp, c = 0;

    printf("Enter size of the  array : ");
    scanf("%d", &n);
    printf("Enter elements in array : ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
        if (a[i] % 2 == 1)
            c++;
    }
    for (i = 0; i < n - 1; i++)
    {

        for (j = 0; j < n - i - 1; j++)
        {
            if (a[j] > a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }

    k = 0;
    j = n - c;

    for (i = 0; i < n; i++)
```

```
    {
        if (a[i] % 2 == 0)
        {
            if (k < n - c)
                b[k++] = a[i];
        }
        else
        {
            if (j < n)
                b[j++] = a[i];
        }
    }

    printf("\narray after sorting even and odd elements separately:\n ");

    for (i = 0; i < n; i++)
    {
        a[i] = b[i];
        printf("%d ", a[i]);
    }
}
```

```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22> cd "c:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22\" ; if ($?) { gcc h2.c -o
 h2 } ; if ($?) { .\h2 }
Enter size of the  array : 5
Enter elements in array : 1
2
3
4
5

array after sorting even and odd elements separately:
 2 4 1 3 5
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

3. Write a program to replace every element in the array with the next greatest element present in the same array.

```
#include <stdio.h>

void nextGreatest(int arr[], int size)
{

    int max_from_right = arr[size - 1];
```

```
    arr[size - 1] = -1;

    for (int i = size - 2; i >= 0; i--)
    {
        int temp = arr[i];

        arr[i] = max_from_right;

        if (max_from_right < temp)
            max_from_right = temp;
    }
}

void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}


    int main()
{
    int arr[] = {16, 17, 4, 3, 5, 2};
    int size = sizeof(arr) / sizeof(arr[0]);
    nextGreatest(arr, size);
    printf("The modified array is: \n");
    printArray(arr, size);
    return (0);
}
```

```
The modified array is:
17 5 5 5 2 -1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

4. WAP to replace every array element by multiplication of previous and next of an n element.

```c
#include <stdio.h>
void newArryPrevNext(int arr1[], int n)
{
    if (n <= 1)
        return;
    int pre_elem = arr1[0];
    arr1[0] = arr1[0] * arr1[1];
    for (int i = 1; i < n - 1; i++)
    {
        int cur_elem = arr1[i];
        arr1[i] = pre_elem * arr1[i + 1];
        pre_elem = cur_elem;
    }
    arr1[n - 1] = pre_elem * arr1[n - 1];
}
int main()
{
    int arr1[] = {1, 2, 3, 4, 5, 6};
    int n = sizeof(arr1) / sizeof(arr1[0]);
    int i = 0;

    printf("The given array is:  \n");
    for (i = 0; i < n; i++)
    {
        printf("%d  ", arr1[i]);
    }
    printf("\n");

    printf("The new array is: \n");
    newArryPrevNext(arr1, n);
    for (int i = 0; i < n; i++)
        printf("%d ", arr1[i]);
    return 0;
}
```

```
The given array is:
1  2  3  4  5  6
The new array is:
2 3 8 15 24 30
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\18-07-22>
```

5. WAP to sort rows of a matrix having m rows and n columns in ascending & columns in descending order.

```c
#include <stdio.h>

void main()
{
    static int array1[10][10], array2[10][10];
    int i, j, k, a, m, n;

    printf("Enter the order of the matrix \n");
    scanf("%d %d", &m, &n);
    printf("Enter co-efficients of the matrix \n");
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            scanf("%d", &array1[i][j]);
            array2[i][j] = array1[i][j];
        }
    }
    printf("The given matrix is \n");
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            printf(" %d", array1[i][j]);
        }
        printf("\n");
    }
    printf("After arranging rows in ascending order\n");
```

```
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            for (k = (j + 1); k < n; ++k)
            {
                if (array1[i][j] > array1[i][k])
                {
                    a = array1[i][j];
                    array1[i][j] = array1[i][k];
                    array1[i][k] = a;
                }
            }
        }
    }
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            printf(" %d", array1[i][j]);
        }
        printf("\n");
    }
    printf("After arranging the columns in descending order \n");
    for (j = 0; j < n; ++j)
    {
        for (i = 0; i < m; ++i)
        {
            for (k = i + 1; k < m; ++k)
            {
                if (array2[i][j] < array2[k][j])
                {
                    a = array2[i][j];
                    array2[i][j] = array2[k][j];
                    array2[k][j] = a;
                }
            }
        }
    }
    for (i = 0; i < m; ++i)
```

```
    {
        for (j = 0; j < n; ++j)
        {
            printf(" %d", array2[i][j]);
        }
        printf("\n");
    }
}
```

```
Enter co-efficients of the matrix
1
2
3
4
5
6
7
8
9
The given matrix is
 1 2 3
 4 5 6
 7 8 9
After arranging rows in ascending order
 1 2 3
 4 5 6
 7 8 9
After arranging the columns in descending order
 7 8 9
 4 5 6
 1 2 3
```